

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-207708

(43)Date of publication of application : 07.08.1998

(51)Int.Cl. G06F 9/38
G05B 19/05

(21)Application number : 09-014363 (71)Applicant : MATSUSHITA ELECTRIC WORKS LTD

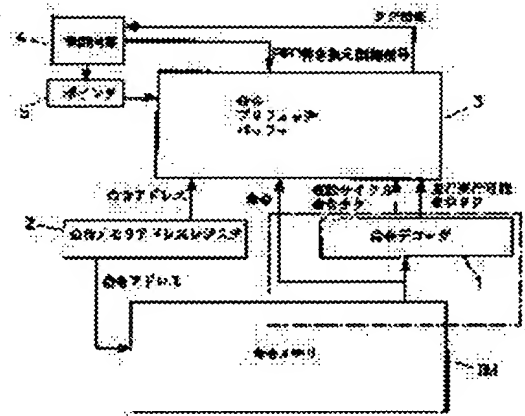
(22)Date of filing : 28.01.1997 (72)Inventor : YABUTA AKIRA
MASUDA TATSUO

(54) PROGRAMMABLE CONTROLLER

(57)Abstract:

PROBLEM TO BE SOLVED: To prevent lowering of a processing speed due to the waiting state of the execution of a following instruction in executing an instruction whose execution cycle is longer than the number of pipe line stages in a programmable controller for executing the high speed processing of the instruction by a pipe line processing.

SOLUTION: This device is provided with a buffer 3 for pre-fetch which temporarily stores an instruction read from an instruction memory and the instruction address, and a buffer pointer 5 which indicates a difference between the leading address of the buffer for pre-fetch and the storage address of the instruction which is being executed in parallel. Then, a plural cycle instruction tag for indicating that the instruction needs plural cycles during the execution of the instruction, and a parallel executable instruction tag for indicating that the instruction can be executed in parallel with the execution of the plural cycle instruction are added to the buffer 3, and the following instruction capable of parallel execution is executed at the time of executing the plural cycle instruction based on those two tags and the buffer pointer 5.



LEGAL STATUS

[Date of request for examination] 15.02.2000

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number] 3405106

[Date of registration] 07.03.2003

[Number of appeal against examiner's decision]

of rejection]

[Date of requesting appeal against examiner's
decision of rejection]

[Date of extinction of right]

(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平10-207708

(43)公開日 平成10年(1998) 8月7日

(51)Int.Cl.⁹

G 0 6 F 9/38

G 0 5 B 19/05

識別記号

3 1 0

F I

G 0 6 F 9/38

G 0 5 B 19/05

3 1 0 F

F

審査請求 未請求 請求項の数8 OL (全 7 頁)

(21)出願番号

特願平9-14363

(22)出願日

平成9年(1997) 1月28日

(71)出願人 000005832

松下電工株式会社

大阪府門真市大字門真1048番地

(72)発明者 薮田 明

大阪府門真市大字門真1048番地 松下電工株式会社内

(72)発明者 増田 達男

大阪府門真市大字門真1048番地 松下電工株式会社内

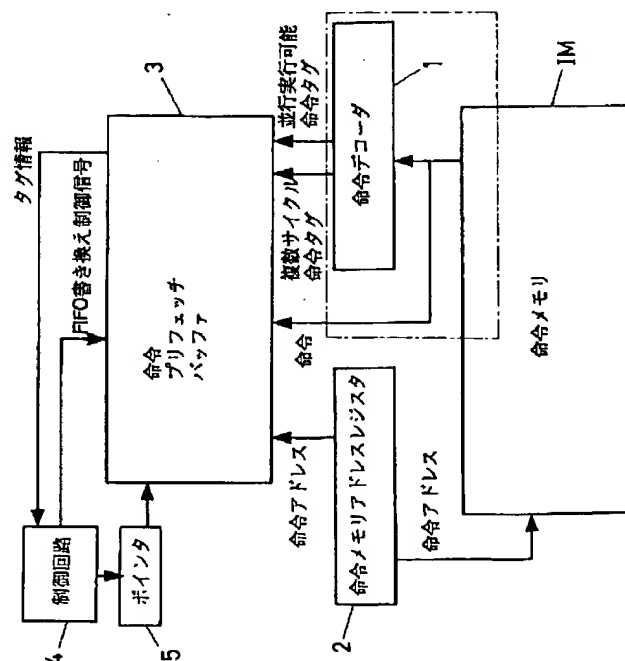
(74)代理人 弁理士 倉田 政彦

(54)【発明の名称】 プログラマブルコントローラ

(57)【要約】

【課題】パイプライン処理によって命令の高速処理を行うプログラマブルコントローラにおいて、実行サイクルがパイプライン段数よりも長い命令の実行時に後続命令の実行が待たせることによる処理速度の低下を防止する。

【解決手段】命令メモリから読み出した命令及びその命令アドレスを一時格納するプリフェッチ用バッファを有し、このプリフェッチ用バッファの先頭アドレスと並列実行中の命令の格納アドレスとの差分を示すバッファポインタを備え、命令実行に複数サイクルを必要とする命令であることを示す複数サイクル命令タグ及び複数サイクル命令実行と並行して実行できる命令であることを示す並列実行可能命令タグを前記バッファに付加し、これら2つのタグと前記バッファポインタに基づいて複数サイクル命令実行時に並列実行可能な後続命令を実行する。



【特許請求の範囲】

【請求項1】 複数のステージをパイプライン実行するパイプライン構造のプログラマブルコントローラであって、命令実行に複数サイクルを必要とする複数サイクル命令実行時の高速処理を実現するために、命令メモリから読み出した命令及びその命令アドレスを一時格納し、実行された命令のみが後続の命令群に書き換えられる先入れ先出しバッファとして機能するプリフェッチ用バッファを有し、このプリフェッチ用バッファの先頭アドレスと並列実行中の命令の格納アドレスとの差分を示すバッファポインタを備え、命令実行に複数サイクルを必要とする命令であることを示す複数サイクル命令タグ及び複数サイクル命令実行と並行して実行できる命令であることを示す並列実行可能命令タグを前記バッファに付加し、これら2つのタグと前記バッファポインタに基づいて複数サイクル命令実行時に並列実行可能な後続命令を実行することを特徴とするプログラマブルコントローラ。

【請求項2】 請求項1において、複数サイクル命令タグは命令のコンパイル時に予め生成されて命令メモリ上の命令に付加されることを特徴とするプログラマブルコントローラ。

【請求項3】 請求項1において、複数サイクル命令タグは命令のプリフェッチ時に命令メモリから読み出された命令に基づいて自動生成してプリフェッチバッファに付加されることを特徴とするプログラマブルコントローラ。

【請求項4】 請求項1において、並列実行可能命令タグは命令のコンパイル時に予め生成されて命令メモリ上の命令に付加されることを特徴とするプログラマブルコントローラ。

【請求項5】 請求項1において、並列実行可能命令タグは命令のプリフェッチ時に命令メモリから読み出された命令に基づいて自動生成してプリフェッチバッファに付加されることを特徴とするプログラマブルコントローラ。

【請求項6】 請求項1乃至5のいずれかにおいて、プリフェッチ用バッファの先頭アドレスの複数サイクル命令タグが1であれば、その命令が終了するまでは前記バッファポインタを1として、プリフェッチ用バッファ上の先頭アドレスの次のアドレスに格納された後続命令を並行処理し、プリフェッチ用バッファの先頭アドレスの複数サイクル命令タグが0であれば、前記バッファポインタを0に戻してプリフェッチ用バッファ上の先頭アドレスの命令を実行することを特徴とするプログラマブルコントローラ。

【請求項7】 請求項6において、前記バッファポインタが1で、プリフェッチ用バッファ上の先頭アドレスの次のアドレスに格納された後続命令についての並列実行可能命令タグが1のときのみ、前記複数サイクル命令

と後続命令の並行処理を行うことにより、実行するとデータが矛盾する命令の実行を阻止することを特徴とするプログラマブルコントローラ。

【請求項8】 請求項7において、前記バッファポインタが0であるときには、プリフェッチ用バッファ上の先頭アドレスの命令の実行後に、先頭アドレスの次のアドレス以降に格納された全命令を順次先頭アドレスまで繰り上げるようにシフトし、前記バッファポインタが1であるときには、プリフェッチ用バッファ上の先頭アドレスの次のアドレスに格納された後続命令についての並列実行可能命令タグが1であれば、該後続命令の実行後に、該後続命令の次のアドレス以降に格納された全命令を順次先頭アドレスの次のアドレスまで繰り上げるようにシフトさせることを特徴とするプログラマブルコントローラ。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、複数のステージをパイプライン実行するパイプライン構造のプログラマブルコントローラに関するものである。

【0002】

【従来の技術】プログラマブルコントローラは産業用装置、機械、FA機器の制御に広く用いられており、対象となる装置の複雑化、高速化に応じて、より多数の入出力信号を高速に処理することが求められている。このため、ビット処理を主とする基本命令と、複数ビットの応用命令処理を行うことのできる専用ハードウェア（プロセッサ）で高速化を実現し、通信処理・周辺処理などを行う汎用マイクロプロセッサと組み合わせてプログラマブルコントローラを構成している。この専用ハードウェア（プロセッサ）の構造として、従来は、以下のような3段パイプライン構造で命令を実行していた。

【0003】

第1ステージ：命令フェッチ

第2ステージ：命令デコード、レジスタフェッチ、算術論理演算、データアドレス計算、分岐先計算、

第3ステージ：メモリアクセス（リード／ライト）、分岐、ビット演算、レジスタ書き込み

【0004】

【発明が解決しようとする課題】上記の3段パイプライン構造では、各ステージのうち一番遅い実行ステージの処理速度で、全体の命令実行速度が決まってしまう。このようなパイプライン処理の高速化のためには、各ステージの処理速度を均等にする必要があるが、この例では命令メモリとデータメモリに同じアクセス時間のメモリを使用すると、第1ステージに対して第3ステージの方がビット演算を行う分、処理時間が必要になるというアンバランスがある。実行速度向上のためには、パイプラインステージをさらに分割して多段にすれば良い。そこで、本発明者らは、専用ハードウェア（プロセッサ）の

命令実行サイクルを以下のような5段に要素分割した、5段パイプライン構造を持ったプログラマブルコントローラを開発した。

【0005】

第1ステージ：命令フェッチ

第2ステージ：命令デコード、レジスタフェッチ

第3ステージ：算術論理演算、データアドレス計算、分岐先計算

第4ステージ：メモリアクセス（リード／ライト）

第5ステージ：分岐、ビット演算、レジスタ書き込み

この5段パイプライン構造を採用すると、従来の3段パイプライン構造のプログラマブルコントローラに比べてパイプラインステージ間のアンバランスが改善され、パイプラインステージ1段あたりの所要時間も短縮されるため、全体の命令実行速度を向上させることができる。この5段パイプライン構造のプログラマブルコントローラの概略構成図を、図3に示す。

【0006】図で、IF (Instruction Fetch) で示される第1ステージは、命令メモリIMから命令レジスタIRへ次に実行する命令を読み込む命令フェッチ処理を行うステージで、命令を格納する命令メモリIMと、プログラムカウンタ制御のためのアドレス計算回路ADDRCALCからの信号を受けて、次に実行する命令が格納された命令メモリIMのアドレスを計数するプログラムカウンタPCとで構成されている。プログラムカウンタPCのアドレス指定に従って命令メモリIMから読み出された命令が格納される命令レジスタIRは、第1ステージIFの実行結果を保存して、次の第2ステージであるIDにその結果を伝える、パイプラインレジスタIF/IDを兼ねている。

【0007】ID (Instruction Decode) で示される第2ステージは、命令デコーダDCによる命令デコード、及び、レジスタファイルRFを構成する複数の汎用レジスタのいずれかより値を取り出すレジスタフェッチ処理を行うステージで、命令のオペレーションコード部を解読するデコーダDCと、複数の汎用レジスタで構成された汎用レジスタファイルRFとで構成されている。汎用レジスタファイルRFには、2つの出力が設けられており、一方の出力はパイプラインレジスタID/EXのS1に接続され、他方の出力はパイプラインレジスタID/EXのS2に接続されている。また、命令デコーダDCで解読された値もパイプラインレジスタID/EXの所定の箇所に格納される。

【0008】次に、EX (Execute) で示される第3ステージは、算術論理演算ユニットALUによつて、算術論理演算またはデータアドレス計算または分岐

アドレス 命令

16 ビット処理命令 (スタート)

17 メモリリード命令 Load R1

18 メモリリード命令 Load R2

先の実効アドレスを計算する分岐先計算を行うステージで、算術論理演算ユニットALUの一方の入力は、パイプラインレジスタID/EXのS1の出力に接続され、他方の入力は、パイプラインレジスタID/EXのS2の出力に接続されている。また、算術論理演算ユニットALUは、パイプラインレジスタID/EXの所定の箇所に格納された、デコードされた命令の値によって制御され、算術論理演算ユニットALUの出力は、パイプラインレジスタEX/MEMのDの箇所に格納される。

【0009】次に、MEM (Memory access) で示される第4ステージは、データメモリDMへのメモリアクセス処理を行うステージで、パイプラインレジスタEX/MEMのDの箇所に格納されていた値は、データメモリDMの所定アドレスのメモリに格納されると共に、パイプラインレジスタMEM/WBの所定箇所に出力される。または、データメモリDMの所定アドレスに格納されていた値がパイプラインレジスタMEM/WBの所定箇所に格納される。

【0010】最後に、WB (Write Back) で示される第5ステージは、ビット演算または汎用レジスタへの書き込み処理または分岐処理を行うステージで、汎用レジスタへの書き込み処理の場合は、パイプラインレジスタMEM/WBの所定箇所に格納されていた値が、レジスタファイルRFの所定の汎用レジスタに格納される。

【0011】図3では、命令メモリIMから命令を取り出す命令フェッチ処理を行う第1ステージIFと、命令デコード処理及び汎用レジスタから値を取り出すレジスタフェッチ処理を行う第2ステージIDと、算術論理演算処理またはデータアドレス演算処理または分岐先の実効アドレス計算処理または分岐条件の判定処理を行う第3ステージEXと、データメモリDMへのメモリアクセス処理または分岐処理を行う第4ステージMEMと、ビット演算処理または前記汎用レジスタへの書き込み処理または分岐処理を行う第5ステージWBの、5つのステージをパイプライン実行する縮小命令型の5段パイプライン構造のプログラマブルコントローラを例示したが、これは説明のために示したのであって、この発明は縮小命令型のプロセッサや5段パイプラインに限定するものではない。

【0012】従来例では、乗算命令・除算命令・微分命令など実行サイクルが規定のパイプライン段数を越える場合、後続の命令実行を停止させている。例えば、以下の例において、乗算命令の後の加算命令と論理積命令は原理的には実行可能であるが、従来例ではこれらの命令の実行を停止させている。

5			
19	乗算命令	M u l t	$R1 * R2 \rightarrow R3$
20	加算命令 (実行可能)	A d d	$R1 + R2 \rightarrow R4$
21	論理積命令 (実行可能)	A n d	$R1 \text{ and } R2 \rightarrow R5$
22	加算命令	A d d	$R3 + R1 \rightarrow R6$
23	加算命令	A d d	$R5 + R6 \rightarrow R7$
24	メモライイト命令	S t o r e	$R7$

【0013】このプログラムでは、アドレス17、18のメモリリード命令によりレジスタR1とR2にメモリのデータを読み込み、アドレス19の乗算命令によりレジスタR1とR2の乗算結果をレジスタR3に格納し、アドレス20の加算命令によりレジスタR1とR2の和をレジスタR4に格納し、アドレス21の論理積命令によりレジスタR1とR2の論理積をレジスタR5に格納し、アドレス22の加算命令によりレジスタR3とR1の和をレジスタR6に格納し、アドレス23の加算命令によりレジスタR5とR6の和をレジスタR7に格納し、アドレス24のメモリライト命令によりレジスタR7のデータをメモリに書き込んでいる。

【0014】したがって、本来ならば、命令アドレスが3番目の乗算命令(アドレス19の $Mult\ R1 * R2 \rightarrow R3$)が実行中は、命令アドレスが4、5番目の命令(アドレス20の加算命令 $Add\ R1 + R2 \rightarrow R4$ とアドレス21の論理積命令 $And\ R1 and R2 \rightarrow R5$)は原理的には実行可能である。なぜなら、乗算命令の結果はレジスタR3に入るので、レジスタR1とR2は値が確定しており、乗算命令の結果として値が変化しないからである。それにもかかわらず、従来例では一律に乗算命令のアドレスで後続命令の実行を停止させている。

【 0 0 1 5 】

【発明が解決しようとする課題】従来のプログラマブルコントローラにおいては、命令のパイプライン処理によって高速処理を実現している。しかし、乗算命令・除算命令・微分命令など一部の命令では、実行サイクルがこのパイプライン段数より長い。このような命令の実行中は、後続命令の実行が待たされるために、結果として処理速度が低下するという問題があった。

【００１６】本発明は、このように、パイプライン処理によって命令の高速処理を行うプログラマブルコントローラにおいて、実行サイクルがパイプライン段数よりも長い命令の実行時に後続命令の実行が待たせることによる処理速度の低下を防止しようとするものである。

【 0 0 1 7 】

【課題を解決するための手段】本発明にあつては、上記の課題を解決するために、図１に示すように、複数のステージをパイプライン実行するパイプライン構造のプログラマブルコントローラであつて、命令実行に複数サイクルを必要とする複数サイクル命令実行時の高速処理を実現するために、命令メモリから読み出した命令及びその命令アドレスを一時格納し、実行された命令のみが後

続の命令群に書き換えられる先入れ先出しバッファとして機能するプリフェッチ用バッファを有し、このプリフェッチ用バッファの先頭アドレスと並列実行中の命令の格納アドレスとの差分を示すバッファポインタを備え、命令実行に複数サイクルを必要とする命令であることを示す複数サイクル命令タグ及び複数サイクル命令実行と並行して実行できる命令であることを示す並列実行可能命令タグを前記バッファに付加し、これら2つのタグと前記バッファポインタに基づいて複数サイクル命令実行時に並列実行可能な後続命令を実行することの特徴とするものであり、請求項に示した構成により、後続の命令で、処理データが確定している命令は、停止させることなく高速処理を実現するものである。

20 【0 0 1 8】

【発明の実施の形態】命令ブリフエッチバッファは、FIFO（先入れ先出し）で深さ8を持つものとする。最初にこのバッファの内容は以下になる。このリストにおいて、各行の書式は「バッファアドレス：命令アドレス 命令」であり、命令の後に付記された1つ目の丸括弧内は実行可能タグの値、2つ目の丸括弧内は複数サイクル命令タグの値である。

	0 : 1 6	スタート		(0)	(0)
	1 : 1 7	Load R1		(0)	(0)
30	2 : 1 8	Load R2		(0)	(0)
	3 : 1 9	Mult R1 * R2→R3		(0)	(1)
	4 : 2 0	Add R1 + R2→R4		(1)	(0)
	5 : 2 1	And R1 and R2→R6		(1)	(0)
	6 : 2 2	Add R3 + R1→R6		(0)	(0)
	7 : 2 3	Add R3 + R6→R7		(0)	(0)
	8 : 2 4	Store R7		(0)	(0)

【0019】この命令配列では、Mul t命令のみが複数サイクルを持つ命令であるため、このMul t命令についてのみ複数サイクル命令タグの値が1となっている。また、この複数サイクルを持つ命令が終了しなくても実行可能な命令にのみ、実行可能タグ：1を付加している。これは、コンパイラが判断して、命令に付加することができる。つまり、コンパイラはMul t命令がレジスタR1とR2の値を書き換えない命令であることをが分かるので、後続命令の加算命令と論理積命令については実行可能であることを判断することができ、コンパイル時に実行可能タグを付加することが可能である。

【0020】また、別の手段として、命令プリフェッチ時に、プリフェッチユニットが複数サイクルを持つ命令を認識して、後続命令についての実行可能タグをプリフ

エッチバッファに付加する。実行可能タグが0の場合は、プリフェッチバッファの先頭に来ない限り、実行されないようにする。

【0021】同じように複数サイクルの命令を示す複数サイクル命令タグは、コンパイラによって命令に付加しておくか、命令プリフェッチ時に、プリフェッチユニットが認識して、プリフェッチバッファに付加する。

【0022】そして、このプリフェッチバッファのどの部分を実行中かを示すために、プリフェッチバッファのポインタを持ち、最初はこのポインタは0を示している。最初の命令（アドレス16）が実行されると、アドレス17以降の命令は、すべて1ずつシフトされて、以下のような構成となる。このリストでも、各行の書式は「バッファアドレス：命令アドレス 命令（実行可能タグ）（複数サイクル命令タグ）」であり、それ以降に登場するリストについても同様である。

【0023】

プリフェッチバッファのポインタ=0

```
0:17 Load R1 (0) (0)
1:18 Load R2 (0) (0)
2:19 Mult R1 * R2→R3 (0) (1)
3:20 Add R1 + R2→R4 (1) (0)
4:21 And R1and R2→R5 (1) (0)
5:22 Add R3 + R1→R6 (0) (0)
6:23 Add R3 + R6→R7 (0) (0)
7:24 Store R7 (0) (0)
```

【0024】次に、アドレス17、18の命令が順次実行された後には、以下のような構成になる。

プリフェッチバッファのポインタ=0

```
0:19 Mult R1 * R2→R3 (0) (1)
1:20 Add R1 + R2→R4 (1) (0)
2:21 And R1and R2→R3 (1) (0)
3:22 Add R3 + R1→R6 (0) (0)
4:23 Add R5 + R6→R7 (0) (0)
5:24 Store R7 (0) (0)
```

このときに複数サイクル命令タグが先頭に来ているので、この命令を実行しながら、後続の並行して実行できる命令を実行ユニットに投入して行く。すなわち、プリフェッチバッファのポインタを1つ進めて処理して行く。

【0025】プリフェッチバッファのポインタ=1

```
0:19 Mult R1 * R2→R3 (0) (1)
1:20 Add R1 + R2→R4 (1) (0)
2:21 And R1and R2→R5 (1) (0)
3:22 Add R3 + R1→R6 (0) (0)
4:23 Add R5 + R6→R7 (0) (0)
5:24 Store R7 (0) (0)
```

ここではアドレス20の命令が実行可能タグ=1なので実行する。ここで、まだ命令19が実行中であれば、プリフェッチバッファのアドレス2以降の命令を選択的に

シフトする。この例では、上記リストのアドレス2～5を下記リストのアドレス1～4に順次シフトする。

【0026】プリフェッチバッファのポインタ=1

```
0:19 Mult R1 * R2→R3 (0) (1)
1:21 And R1and R2→R5 (1) (0)
2:22 Add R3 + R1→R6 (0) (0)
3:23 Add R5 + R6→R7 (0) (0)
4:24 Store R7 (0) (0)
```

ここでは、アドレス21の命令が実行可能タグ=1なので実行する。ここで、まだ命令19が実行中であれば、同じく、プリフェッチバッファのアドレス2以降の命令を選択的にシフトする。この例では、上記リストのアドレス2～4を下記リストのアドレス1～3に順次シフトする。

【0027】プリフェッチバッファのポインタ=1

```
0:19 Mult R1 * R2→R3 (0) (1)
1:22 Add R3 + R1→R6 (0) (0)
2:23 Add R5 + R6→R7 (0) (0)
3:24 Store R7 (0) (0)
```

ここではアドレス22の命令が実行可能タグ=0なので実行せずに、アドレス19の命令実行が終わるのを待つ。つまり、アドレス22の命令は乗算命令の結果であるレジスタR3の値を使用するので、実行可能タグが0となっているのである。アドレス19の命令が実行終了したら、プリフェッチバッファのポインタを0に戻し、通常の実行サイクルに戻る。

【0028】プリフェッチバッファのポインタ=0

```
0:22 Add R3 + R1→R6 (0) (0)
1:23 Add R5 + R6→R7 (0) (0)
2:24 Store R7 (0) (0)
```

ここでは表示を省略しているが、これら実行サイクルの間に、アドレス25以降の命令も継続してプリフェッチされて行くことは当然である。

【0029】

【実施例】本発明の1つの実施例として、図2にプリフェッチバッファの深さが8の場合を示す。この深さは任意に設定できることは言うまでもない。命令がメモリからプリフェッチされると、まずバッファ7にその命令アドレスと命令が格納される。次の命令がプリフェッチされると先に取り込まれた命令はバッファ6に移され、バッファ7に新たな命令が入る。このように命令が取り込まれて行き、バッファ7～0まで命令が埋められた後に、バッファ0の命令から実行されて行く。この後の動作は、上述の通りである。

【0030】このとき、複数サイクル命令タグと並列実行可能タグは、命令コンパイル時に決める方式であれば、各タグは命令に組み込まれているので、命令メモリから取り込むだけで、バッファに格納される。つまり、本来の命令語長がnビットであれば、コンパイル後の命令語長は複数サイクル命令タグと並列実行可能タグを付

加して $(n+2)$ ビットとなっており、その命令語長で命令メモリに格納されているから、命令メモリから読み出してバッファに書き込むだけで良い。また、別の手段として、プリフェッチ時にこれら2種類のタグを決める方式の場合には、命令プリフェッチブロックに命令デコーダを配して、命令メモリから読み出した命令について、これらタグを自動判定し、タグを付加した状態でバッファに格納して行く。

【0031】命令プリフェッチバッファと周辺回路の一例を図1に示す。命令メモリIMから取り込まれる命令は命令デコーダ1を通じてタグが生成される。命令アドレスはメモリアクセス（プリフェッチ）の際に用いる命令メモリアドレスレジスタ2の内容を取り込む。ここで、命令コンパイル時にタグがすでに命令に付加されている場合には、上図の命令デコーダ1（図1の一点鎖線で囲まれた部分）は不要であり、これを通さずに命令そのものに付加されているタグがそのまま命令プリフェッチバッファ3に取り込まれる。制御回路4は命令プリフェッチバッファ3のタグ情報をもとに、プリフェッチバッファアドレスポインタ5の値を書き換え、また、FIFO

IFOの書き換えを制御することによって、並列実行の制御を行う。

【0032】

【発明の効果】本発明によれば、命令のパイプライン処理によって高速処理を実現しているプログラマブルコントローラにおいては、乗算命令・除算命令・微分命令などのように、実行サイクルがパイプライン段数より長い一部の命令の実行中にも、後続命令の実行が待たされることを防止し、結果として処理速度が低下することを防止できるという効果がある。

【図面の簡単な説明】

【図1】本発明に用いる命令プリフェッチバッファと周辺回路を示すブロック回路図である。

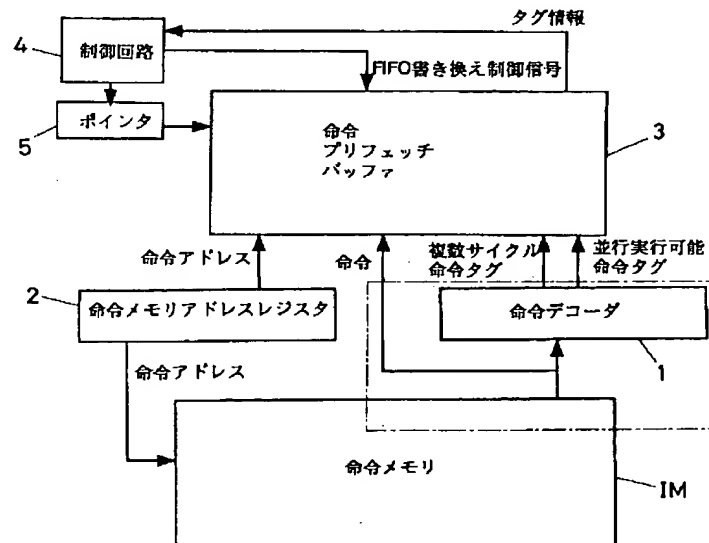
【図2】本発明に用いる命令プリフェッチバッファの構造を示す説明図である。

【図3】従来のパイプライン構造のプログラマブルコントローラのブロック回路図である。

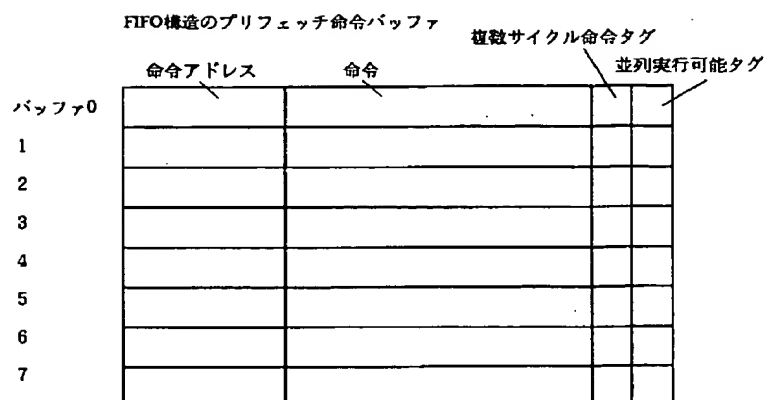
【符号の説明】

3 命令プリフェッチバッファ
5 ポインタ

【図1】



【図2】



【図3】

